Universität Potsdam
Humanwissenschaftliche Fakultät
Pogrammiersprache
Projektreporting zu ‘Interactive Fiction’
Daryna Ivanova
Matrikelnummer: 804197

Die Aufgabe des Projektes war ein textbasiertes Spiel zu implementieren, bei dem der Benutzer sich zwischen den Situationen durch Texteingaben bewegt.

Mein Projekt ist so strukturiert, dass die meisten logischen Schritte in der Klasse *Character* durchgeführt werden. Allerdings habe ich versucht, das Program möglichst objektorientiert zu halten, so dass es möglichst unkompliziert erweitert und bearbeitet werden könnte.

Das Programm kann aus der main-Datei aufgerufen werden. Dabei werden die ersten Szenenbeschreibungen angezeigt und die *Chacter* Klasse instanziiert. Innerhalb einer while-Schleife wird der Benutzer nach einer Texteingabe solange gefragt, bis das Spiel beendet ist.

Die Klasse *Scenes* liest die txt-Dateien mit den Szenenbeschreibungen und gibt sie bei der Klasseninitialisierung in dem Terminal aus.

FirstRiddle und *SecondRiddle* entsprechen den Rätseln. Sie erben von einer abstrakten Klasse *ABCRiddle* und haben zwei abstrakten Methoden ‘random_data’, wo die Daten für die Rätsel vorbereitet und ausgewählt werden, und ‘riddle’, wo die eigentliche Logik eines Rätsels programmiert ist.

Die *Character* Klasse repräsentiert den Spieler und hat solche Charakteristika wie ‘current_location’, ‘logic_dict’, ‘bag’ und ‘time_for_riddle’. Außerdem beinhaltet diese Klasse die Methoden, die allen in dem Spiel möglichen Operationen entsprechen. Diese werden nach einer Commandoeingabe in der Methode *act()* koordiniert.

Die Idee ist, dass beim Erstellen eines Objektes der *Character* Klasse, einen String ‘Potsdam Hbf’ übergeben werden muss. Das ist der erste Zustand, der dem Attributen ’current_location’ Inder Klasse *Character* entspricht. ‘current_location’ wird zu einer Instanz der *Location* - Klasse umgewandelt. Infolgedessen kann ‘current_location’ mithilfe ‘change_location’- Methode im Laufe des Spiels mehrfach geändert werden.

‘bag’ ist ein Dictionary, das je nach Bewältigung der thematischen Szenen bearbeitet werden kann. So wird es z.B. nach dem Rätsellösung beim Fahrkartenautomat (‘Ticket machine’) von {"money": 20, "ticket": False, "book": False} zu {"money": 15, "ticket": True, "book": False} angepasst.

‘logic_dict’ bezieht die Zustände mit deren zulässigen Befehlen ein. Ich habe dafür ein Dictionary ausgewählt. Das erlaubt dann die Überprüfung, ob die Texteingaben des Benutzers in einer Liste mit den möglichen Spielbefehlen zu finden sind. Darüber hinaus ist ‘logic_dict’ auch bei der Eingabe der Befehlen, die nicht in einem bestimmten Zustand durchführbar sind, hilfreich. Die Methode *get_message()* konstruiert eine standardisierte Ausgabe, je nach aktuellem Zustand/ Szene. Zum Beispiel, wenn der Benutzer das Commando ‘get to golm’ vor einer der Zugszenen eingibt, wird so eine Meldung erstellt:

```
What do you want to do?
⏩ get to golm
 You can't do it here, but here are the actions you can do: ['inspect
bag', 'get train', 'exit']
```

’time_for_riddle’ wird nach der Bewältigung des ersten Rätsels, die verbrauchte Zeit als Integer-Wert übernehmen. Diese Zeit ist dann beim Ausführen von ‘get train’-Commando entscheidend, denn sie bestimmt welche Wendung das Spiel nimmt.